دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

کارشناسی ارشد علوم کامپیوتر گرایش داده کاوی

پروژه شماره ۳ درس داده کاوی

نگارش

حدیث حق‌شناس جزی

استاد راهنما

مهدی قطعی

استاد مشاور

بهنام یوسفی مهر

آبان **۱۴۰۱**

چکیده

در این پروژه مراحل پیش پردازش داده و مهندسی ویژگی ها بر روی یک دیتاست شامل ویژگی های ظاهری و وزنی و قیمت و اندازه **۵۴** هزار الماس انجام شده است. در طی این عملیات **۱۱** ستون دیتاست به **۲۵** ستون تبدیل شده و سپس رگرسیون خطی روی داده ها فیت میشود. در این پروژه از RFE در قسمت استخراج ویژگی ها استفاده شده است.

۱ فصل اول

مقدمه

در این پروژه سعی شده‌است که یک دیتاست مناسب برای پیش پردازش و مهندسی ویژگی ها انتخاب شود. سپس به معرفی دیتاست، انجام تحلیل های آماری داده های مربوطه جهت پاکسازی داده و انجام روش های مختلف انتخاب و استخراج ویژگی و تجزیه و تحلیل داده های اکتشافی و استفاده از مدل رگرسیون بر روی داده ها می‌پردازیم.

همچنین برای نتایج بهتر مجددا بعضی از اقدام ها را تکرار کرده و به یک مدل بهینه نسبت به مدل اولیه خواهیم رسید.

۲ پیش پردازش

2-1 دیتاست diamond prices 2022

دیتاست الماس شامل 11 ستون و حدود 54 هزار سطر میباشد و مواردی مانند مدل تراش، قیراط، رنگ، شفافیت، اندازه، قیمت و انواع داده های دیگر را میتوان در آن مشاهده کرد. زین پس بیش از آنکه به توضیح واضحات بپردازیم، نتیجه کد های موجود بر دیتاست را نمایش داده و به توضیح مختصر هر کد خواهیم پرداخت.

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...
53938	53939	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64
53940	53941	0.71	Premium	E	SI1	60.5	55.0	2756	5.79	5.74	3.49
53941	53942	0.71	Premium	F	SI1	59.8	62.0	2756	5.74	5.73	3.43
53942	53943	0.70	Very Good	E	VS2	60.5	59.0	2757	5.71	5.76	3.47

53943 rows × 11 columns

2-2 بررسی دیتاست

```
df_dim.drop('Unnamed: 0', axis=1, inplace=True)
```

```
df_dim.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53943 entries, 0 to 53942
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    53943 non-null  float64
 1   cut      53943 non-null  object
 2   color    53943 non-null  object
 3   clarity  53943 non-null  object
 4   depth    53943 non-null  float64
 5   table    53943 non-null  float64
 6   price    53943 non-null  int64
 7   x        53943 non-null  float64
 8   y        53943 non-null  float64
 9   z        53943 non-null  float64
dtypes: float64(6), int64(1), object(3)
memory usage: 4.1+ MB
```

مطابق شکل اولیه ستون "unnamed: 0" یک ستون اضافی میباشد لذا در ادامه با حذف این ستون مدل داده هارا بررسی میکنیم. مطابق اطلاعات دیتاست، 3 مدل داده صحیح، اعشاری و کیفی در این دیتاست موجود میباشد.

In [6]:
```
df_dim.describe()
```

Out[6]:

	carat	depth	table	price	x	y	z
count	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000	53943.000000
mean	0.797935	61.749322	57.457251	3932.734294	5.731158	5.734526	3.538730
std	0.473999	1.432626	2.234549	3989.338447	1.121730	1.142103	0.705679
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.000000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

طبق این جدول اطلاعات میانگین، پراکندگی، مینیمم و ماکسیمم و چارک داده های عددی مشخص شده است.

```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
num_df = df_dim.select_dtypes(include=numerics)
num_df
```

	carat	depth	table	price	x	y	z
0	0.23	61.5	55.0	326	3.95	3.98	2.43
1	0.21	59.8	61.0	326	3.89	3.84	2.31
2	0.23	56.9	65.0	327	4.05	4.07	2.31
3	0.29	62.4	58.0	334	4.20	4.23	2.63
4	0.31	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...
53938	0.86	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	62.2	55.0	2757	5.83	5.87	3.64
53940	0.71	60.5	55.0	2756	5.79	5.74	3.49
53941	0.71	59.8	62.0	2756	5.74	5.73	3.43
53942	0.70	60.5	59.0	2757	5.71	5.76	3.47

53943 rows × 7 columns

برای انجام پردازش های مختلف قبل از تبدیل کردن داده های کیفی به عددی، این پردازش ها را در دو دسته دیتاهای عددی و کیفی بررسی میکنیم.

In [9]:
```
categ = ['object']
categ_df = df_dim.select_dtypes(include=categ)
categ_df
```

Out[9]:

	cut	color	clarity
0	Ideal	E	SI2
1	Premium	E	SI1
2	Good	E	VS1
3	Premium	I	VS2
4	Good	J	SI2
...	...	...	...
53938	Premium	H	SI2
53939	Ideal	D	SI2
53940	Premium	E	SI1
53941	Premium	F	SI1
53942	Very Good	E	VS2

53943 rows × 3 columns

داده های کیفی شامل 3 ستون بالا(مدل برش، رنگ، شفافیت) میباشند.

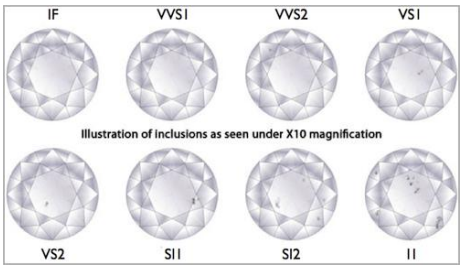
```
In [10]: df_dim.cut.value_counts()

Out[10]: Ideal        21551
         Premium      13793
         Very Good    12083
         Good          4906
         Fair          1610
         Name: cut, dtype: int64
```

```
In [11]: df_dim.color.value_counts()

Out[11]: G    11292
         E     9799
         F     9543
         H     8304
         D     6775
         I     5422
         J     2808
         Name: color, dtype: int64
```

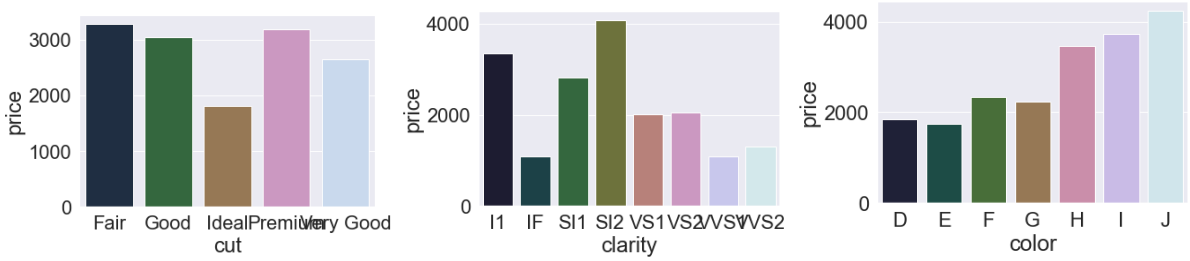
```
In [12]: df_dim.clarity.value_counts()

Out[12]: SI1     13067
         VS2     12259
         SI2      9194
         VS1      8171
         VVS2     5066
         VVS1     3655
         IF       1790
         I1        741
         Name: clarity, dtype: int64
```

ابتدا به بررسی فراوانی هر مجموعه در فیاچر های کیفی میپردازیم. طبق این اعداد تعداد داده هایی که در 2 دسته جداگانه I1 و fair قرار میگیرند بسیار پایین هستند.

در پایین میتوان مشاهده کرد که هر دسته گویای چه مواردی میباشد :



3 مهندسی ویژگی ها و تجزیه و تحلیل داده های اکتشافی



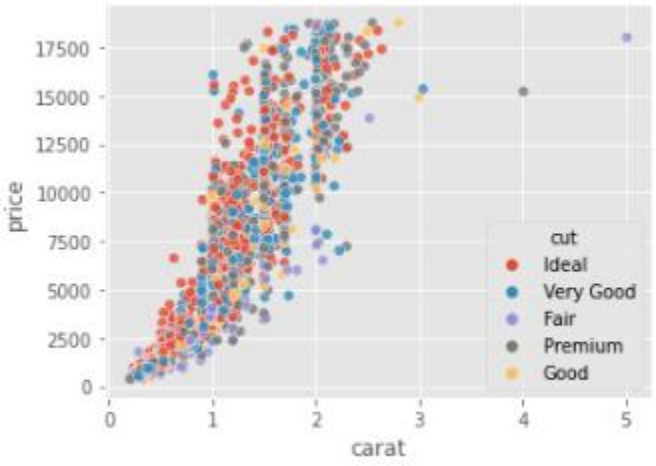
اکنون نمودارهای توزیع هر دسته از ستون های کیفی را نسبت به میانه قیمت ها میکشیم. طبق این نمودار ها در رابطه مدل برش و قیمت الماس ها نمیتوانیم اظهار نظری داشته باشیم زیرا گمان میبردیم که هرچه برش دقیق تر باشد قیمت الماس بیشتر میشود اما چنین نیست. اما به این دلیل که این نمودار بر حسب میانه قیمت کشیده شده است نمیتوانیم اظهار کنیم که رابطه مدل برش و قیمت الماس ها دقیقا از چنین نموداری پیروی میکند.

درباره وضوح الماس ها و رنگ آنها این موضوع شفاف تر میباشد. هرچه الماس رنگ بیشتری داشته باشد(طبق عکس رنگی الماس ها) قیمت آن افزایش دارد. همچنین در نمودار دیگرتوزیع وضوح نسبت به قیمت نشان داده شده است که در دسته های آخر تقریبا توزیع یکسانی داریم.

پس از این بهتر است سراغ نمودار همبستگی برویم. تا ارتباط میان داده های عددی را بررسی کنیم.

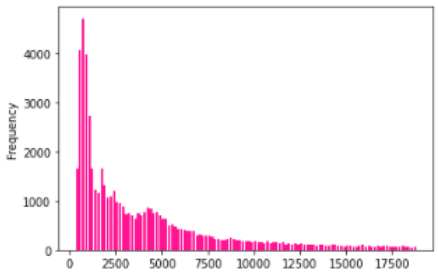
میان

همانطور که مشاهده میشود همبستگی مثبتی را میان carat و price مشاهده میکنیم، در ادامه این فرض نمودار نقطه ای قیراط نسبت به قیمت با تفضیل نوع برش میکشیم تا رابطه این دو فیاچر را بهتر مشاهده کنیم:

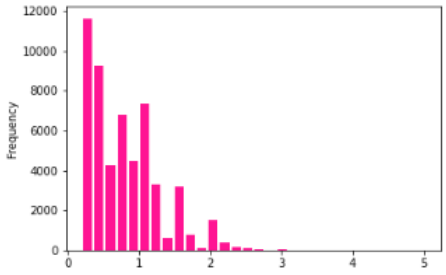


در این نمودار مشاهده میشود که هرچه قیراط الماس بیشتر شود قیمت الماس نیز رشد میکند. از طرفی با توجه به رنگ های متفاوت نمودار مشاهده میکنیم که رنگ قرمز(برش ایده آل) در هر بازه ثابت قیراط شامل قیمت های بالاتری میباشد و قسمت های پایین تر نمودار را رنگ آبی پوشانده است (برش معمولی)

```
In [66]: import matplotlib as plt
         df['price'].plot(kind = 'hist', bins= 111 , rwidth=0.8 , color='deeppink');
```




```
In [77]: df['carat'].plot(kind = 'hist', bins= 30 , rwidth=0.8 , color='deeppink');
```



به دلیل مهم بودن این دو ویژگی نمودار های هیستوگرام رسم کرده ایم تا میزان پراکندگی و وضعیت توزیع این فیاچر ها بهتر دیده شوند.

```
In [13]: categorical_features=[feature for feature in df_dim.columns if df_dim[feature].dtypes=='O']
         print('number of categorical variables:',len(categorical_features))

         number of categorical variables: 3
```

```
In [14]: df_dim[categorical_features].head()
```

Out[14]:

	cut	color	clarity
0	Ideal	E	SI2
1	Premium	E	SI1
2	Good	E	VS1
3	Premium	I	VS2
4	Good	J	SI2

```
In [15]: categorical_features
Out[15]: ['cut', 'color', 'clarity']
```

سپس در جهت مهندسی ویژگی ها تلاش بر این است که داده های کیفی جهت بررسی بیشتر به داده عددی تبدیل شود.

```
In [26]: from sklearn.compose import make_column_transformer
         from sklearn.preprocessing import OneHotEncoder

         onehotencoder = OneHotEncoder()

         transformer = make_column_transformer((OneHotEncoder(), ['cut','color','clarity']), remainder='passthrough')
         transformed = transformer.fit_transform(df_dim)
         transformed_df = pd.DataFrame( transformed, columns=transformer.get_feature_names())

         df = transformed_df
         df
```

Out[26]:

	onehotencoder__x0_Fair	onehotencoder__x0_Good	onehotencoder__x0_Ideal	onehotencoder__x0_Premium	onehotencoder__x0_Very Good	onehotencoder__x1_
0	0.0	0.0	1.0	0.0	0.0	0.
1	0.0	0.0	0.0	1.0	0.0	0.
2	0.0	1.0	0.0	0.0	0.0	0.
3	0.0	0.0	0.0	1.0	0.0	0.
4	0.0	1.0	0.0	0.0	0.0	0.
...	...	...	...	...	...	.
53938	0.0	0.0	0.0	1.0	0.0	0.
53939	0.0	0.0	1.0	0.0	0.0	1.
53940	0.0	0.0	0.0	1.0	0.0	0.
53941	0.0	0.0	0.0	1.0	0.0	0.
53942	0.0	0.0	0.0	0.0	1.0	0.

53943 rows × 27 columns

هر ستون به تعداد مدل داده های موجود به ستون های 0-1 تبدیل میشود.

In [29]:
```
df = df.rename(columns = ({'onehotencoder__x0_Fair':'fair', 'onehotencoder__x0_Good':'good',
       'onehotencoder__x0_Ideal': 'ideal', 'onehotencoder__x0_Premium':'premium',
       'onehotencoder__x0_Very Good':'verygood', 'onehotencoder__x1_D':'D',
       'onehotencoder__x1_E':'E', 'onehotencoder__x1_F':'F', 'onehotencoder__x1_G':'G',
       'onehotencoder__x1_H':'H', 'onehotencoder__x1_I':'I', 'onehotencoder__x1_J': 'J',
       'onehotencoder__x2_I1':'I1', 'onehotencoder__x2_IF':'IF', 'onehotencoder__x2_SI1':'SI1',
       'onehotencoder__x2_SI2':'SI2', 'onehotencoder__x2_VS1':'VS1',
       'onehotencoder__x2_VS2':'VS2', 'onehotencoder__x2_VVS1':'VVS1',
       'onehotencoder__x2_VVS2':'VVS2'}), inplace = False)

#'Unnamed: 0':'count'
df.head()
```

Out[29]:

	fair	good	ideal	premium	verygood	D	E	F	G	H	...	VS2	VVS1	VVS2	carat	depth	table	price	x	y	z
0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.23	61.5	55.0	326.0	3.95	3.98	2.43
1	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.21	59.8	61.0	326.0	3.89	3.84	2.31
2	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.23	56.9	65.0	327.0	4.05	4.07	2.31
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.29	62.4	58.0	334.0	4.20	4.23	2.63
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.31	63.3	58.0	335.0	4.34	4.35	2.75

5 rows × 27 columns

به دلیل سختی اسم ستون ها به تغییر اسم ستون ها میپردازیم. سپس به دلیل تعداد بالای ستون ها و سختی محاسبه به حذف 2 ستون با تعداد پایین تر (در بالا مشاهده کردیم) میپردازیم. و دوباره دیتاست را نمایش میدهیم.

In [33]:
```
df = DF.copy()
```

In [34]:
```
df.drop('fair', axis=1, inplace=True)
```

In [35]:
```
df.drop('I1', axis=1, inplace=True)
```

In [36]:
```
df
```

Out[36]:

	good	ideal	premium	verygood	D	E	F	G	H	I	...	VS2	VVS1	VVS2	carat	depth	table	price	x	y	z
0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.23	61.5	55.0	326.0	3.95	3.98	2.43
1	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.21	59.8	61.0	326.0	3.89	3.84	2.31
2	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.23	56.9	65.0	327.0	4.05	4.07	2.31
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	0.0	0.29	62.4	58.0	334.0	4.20	4.23	2.63
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.31	63.3	58.0	335.0	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
53938	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.86	61.0	58.0	2757.0	6.15	6.12	3.74
53939	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.75	62.2	55.0	2757.0	5.83	5.87	3.64
53940	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.71	60.5	55.0	2756.0	5.79	5.74	3.49
53941	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.71	59.8	62.0	2756.0	5.74	5.73	3.43
53942	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.70	60.5	59.0	2757.0	5.71	5.76	3.47

53943 rows × 25 columns

اکنون 25 ستون داریم.

```
In [50]: normalized_df=(df-df.mean())/df.std()
```

```
In [52]: y = normalized_df['price']
X = normalized_df[['good','ideal','premium','verygood','D','E','F','G','H','I','IF','SI1','SI2','VS1','VS2','VVS1','VVS2','carat'
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(43154, 20) (10789, 20)
(43154,) (10789,)
```

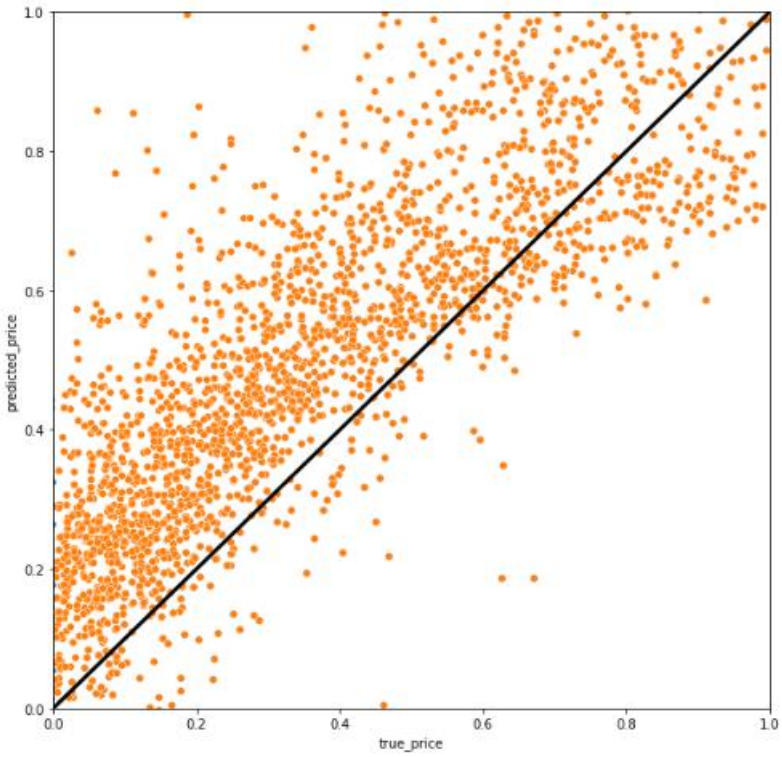
```
In [53]: model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mae_lr = mean_absolute_error(y_test, y_pred)
mape_lr = mape(y_test, y_pred)

print(f'the mean absolute error for linear regression is {round(mae_lr, 2)}')
print(f'the mean absolute percentage error for linear regression is {round(mape_lr, 2)}')
```

```
the mean absolute error for linear regression is 0.2
the mean absolute percentage error for linear regression is 1.34
```

در اینجا داده هارا نرمال سازی میکنیم و مدل رگرسیون را پیاده میکنیم.

اعداد بالا نمایش دهنده MSE و درصد خطای کمترین مربعات میباشد.

مدل رگرسیون را نمایش میدهیم:



In [70]:
```
predictions = rfe.predict(X_test)
```

In [72]:
```
from operator import itemgetter
features = X_train.columns.to_list()
for x, y in (sorted(zip(rfe.ranking_ , features), key=itemgetter(0))):
    print(x, y)
```

```
1 D
1 E
1 F
1 IF
1 SI1
1 SI2
1 VVS1
1 VVS2
1 carat
1 depth
2 G
3 VS2
4 VS1
5 I
6 H
7 table
8 ideal
9 good
10 verygood
11 premium
```

سپس توسط الگوریتم RFE به انتخاب مهمترین ستون ها میپردازیم. سپس دوباره مدل رگرسیون را اجرا میکنیم.

In [77]:
```
y = normalized_df['price']
X = normalized_df[['D','E','F','IF','SI1','SI2','VVS1','VVS2','carat', 'depth']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
print(X_train.shape, X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(43154, 10) (10789, 10)
(43154,) (10789,)
```

In [79]:
```
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mae_lr = mean_absolute_error(y_test, y_pred)
mape_lr = mape(y_test, y_pred)

print(f'the mean absolute error for linear regression is {round(mae_lr, 2)}')
print(f'the mean absolute percentage error for linear regression is {round(mape_lr, 2)}')
```

```
the mean absolute error for linear regression is 0.22
the mean absolute percentage error for linear regression is 1.3
```

مشاهده میکنیم که درصد خطا کاهش دارد و همچنین با وجود حذف 15 ستون و تنها با به کارگیری 10 ستون میزان خطا تغییر جزئی داشته است و این یعنی الگوریتم به خوبی در حذف ستون های بی فایده کمک کرده است.

نتیجه گیری

با توجه به بررسی دیتاست تمام اطلاعات مربوط به این الماس ها مخصوصا اطلاعات برش مفید نیستند لذا با به کارگیری الگوریتم های مرتبط حذف شده اند اما در میان این اطلاعات متوجه شدیم که الماس هایی با رنگ های تیره تر و قیراط بالاتر قیمت بیشتری را دارند.

منابع و مراجع:

- https://www.kaggle.com/datasets/nancyalaswad90/diamonds-prices

پایان